Identifying Multiple Optima in Aerodynamic Design Spaces

D.J. Poole [*], C.B. Allen [†], T.C.S. Rendall[‡]

Department of Aerospace Engineering, University of Bristol, Bristol, BS8 1TR, U.K.

Parallel niching optimization algorithms are developed and applied to a multimodal aerodynamic optimization case to identify multiple optima in the design space. Previous work by the authors has presented niching optimization algorithms that use differential evolution with feasible selection as a basis that can identify multiple optima in constrained search spaces, which is necessary for aerodynamic optimization. In this paper, these algorithms are further developed for application to aerodynamic optimization by reformulating each to provide a parallel decomposition of the objective function evaluation at each iteration of the optimization process. These algorithms are tested on an analytical optimization problem. The parallel, constrained form of both local nearest-neighbourhood and local crowding are shown to be the best performing algorithms with a 99% confidence level. A variation on the AIAA ADODG case 6 multimodal wing optimization case is also studied. A multi-start gradient-based approach is used to show multimodality of the design space. The local nearest-neighbourhood-based algorithm is then applied to the aerodynamic optimization case and is able to successfully identify two minima in the design space, with one being close to the global minimum and one in a different part of the design space, which is a local minimum.

I. Introduction and Background

Engineering optimization typically involves changing a number of design parameters to improve some form of metric (also called an objective), which is normally measured by performing a computational analysis. However, due to the nature of design, constraints on the form that the solution can take often exist and the final optimum solution must satisfy all of these constraints; a solution is said to be feasible if it satisfies the constraints. These types of problems are termed constrained numerical optimization problems (CNOPs). The trend in engineering optimization is towards higher-fidelity simulation methods being integrated into the process. An example of this is a computational fluid dynamics (CFD) simulation being used to determine objective and constraint values in aerodynamic shape optimization (ASO) [1–5].

Nature-inspired meta-heuristics have become commonplace when solving traditional unconstrained numerical optimization problems (UNOPs). Typically these approaches use a population of individuals who cooperate together in search of the optimal solution; evolutionary algorithms (EAs)–such as genetic algorithms (GAs) [6] or differential evolution (DE) [7]–and swarm intelligence algorithms (SIAs)–such as particle swarm optimization (PSO), artificial bee colony system [8] or firefly algorithm [9]–are popular nature-inspired approaches. When solving a CNOP, nature-inspired algorithms typically have to be coupled with a constraint handling method, such that a feasible solution can be found. These methods tend to be *ad hoc*, but have led to the successful application of nature-inspired algorithms to solving CNOPS; see [10–17], for example.

Whether solving a UNOP or a CNOP, generally the goal is to locate the overall best (feasible) solution within the search space, however, it may also be desirable to locate all of the optimal solutions (whether these are global or local). For example, the so-called "*second Toyota paradox*"[18] showed that by considering many different design candidates through the process, rather than converging on one design quickly, can result in an overall better and more cost-effective product. In the context of optimization, this type of approach can

[*]Senior Teaching Associate. Email: d.j.poole@bristol.ac.uk

[†]Professor of Computational Aerodynamics. Email: c.b.allen@bristol.ac.uk

[‡]Lecturer. Email: thomas.rendall@bristol.ac.uk

be defined as multimodal optimization, where for a given design space of an objective function, the goal is not to simply locate the best available solution within that space, but the set of individually identifiable optimal solutions. These may either be all equally optimal (i.e. global optima) or some may be better than others but each solution is better than another in its vicinity (i.e. local optima). The algorithms developed and commonly employed for performing such optimizations are called "niching" methods, examples of which are crowding [19, 20], fitness sharing [6, 20, 21], clearing [22], speciation [23, 24] dynamic archives [25], probabilistic selection [26] and local neighbourhoods [27, 28]. A full review is presented by Li *et al.*[29].

Aerodynamic shape optimization (ASO) is the process of finding the shape (an aerofoil in two dimensions or generally a wing or rotor blade in three dimensions) that optimizes a performance quantity, typically drag, subject to force, moment and geometric constraints, where the objective and constraints are commonly evaluated using a numerical method. Much of the work to date has been involved with finding the single best optimum solution within various design spaces, however, during the design process, identification of other optimal designs may have positive impacts on cost and performance. The identification of multiple optima within a design space clearly requires that the space be multimodal, which for aerodynamic optimization is very problem dependent. For aerofoil optimization, it has been suggested that ASO problems exist that are unimodal [30] and multimodal[31, 32], while the same is true of wings exhibiting unimodality [33] and multimodality [30]. Recently, results of the optimization of a rectangular wing in inviscid subcritical flow (this is case 6 of the AIAA ADODG [34]) have been presented investigating the degree of multimodality in that case [35–38]. However, while some aerodynamic optimization problems are multimodal, there is little effort to attempt to identify and characterise these. Chernukhin and Zingg [30] did characterise the optimal solutions of a number of problems by using a multi-start gradient-based approach. Similar approaches were performed by Bons *at al.* [36], and Streuber and Zingg [38]. The current authors performed similar work by running a heuristic global optimizer multiple times on the same problem to analyse whether it converged to different solutions each time [37].

The application of conventional nature-inspired methods to aerodynamic optimization has been significant (see, for example [4, 39–44]), however the use of niching methods is much less despite the clear motivation to locate multiple optima in aerodynamic problems. The main application of such approaches to date is from Obayashi *et al.* [45] who used niching techniques to locate the pareto front of a multi-objective wing design problem. Probably the main reason for the lack of use is the lack of fundamental development of niching methods for performing constrained optimization; aerodynamic optimization problems are almost invariably constrained. In a recent study on niching methods [29], it was stated that:
"*there lacks a systematic study on how existing niching methods, largely designed for unconstrained optimization, should cope with constraints*".

As such, the authors have recently developed state-of-the-art algorithms for locating multiple optima within constrained design spaces where DE-based niching methods that use feasible selection of individuals were developed. The work in this paper is concerned with the application of those sophisticated approaches to the problem of aerodynamic optimization. The primary issue regarding the application of such approaches to aerodynamic optimization is the cost associated with evaluating the objective function, which is large. Hence, parallelisation of objective function evaluations is necessary to permit reasonable run times of population-based algorithms. This paper will first consider parallel decomposition of the algorithms previously developed by the authors. This work will also consider the AIAA ADODG case 6 as an aerodynamic optimization example.

II. Multimodal Optimization Definition

A general definition of a CNOP and multimodal optimization problem is presented in this section. A CNOP is described by equation 1.

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{S}\in\Re^D} && f(\mathbf{x}) \\ &\text{subject to} && g_j(\mathbf{x}) \le 0 \quad , \quad j = 1,\dots,p \\ & && h_j(\mathbf{x}) = 0 \quad , \quad j = 1+p,\dots,m \end{aligned} \tag{1}$$

In equation 1, $\mathbf{x}$ is the solution vector $[x^1, x^2, \dots, x^D]^T$ where each element of the vector is a design variable in a problem containing D design variables; $f(\mathbf{x})$ is the value of the objective function for the given solution vector; $g_j(\mathbf{x})$ represents the j-th inequality constraint of a total of p inequality constraints; $h_j(\mathbf{x})$

represents the $(j-p)$-th equality constraint of a total of $m-p$ inequality constraints where m is the total number of constraints; $\mathcal{S}$ is the bounded region of $\Re^D$ where the solution exists, which is a D-orthotope defined by an upper bound in the k-th dimension, U^k, and a lower bound, L^k, where $k = \{1, 2, \ldots, D\}$. The feasible region, $\mathcal{F}$, defines the set of all feasible solutions. In this work, it is only the location of all global (or close to global) optima that are considered. The multimodal solution of a CNOP is therefore the set containing N_g global optima, $\mathcal{X} = \{\mathbf{x}^{*1}, \ldots, \mathbf{x}^{*N_g}\}$ that minimise f in $\mathcal{F}$, hence:

$$f(\mathbf{x}^{*1}) = \cdots = f(\mathbf{x}^{*N_g}) < f(\mathbf{x}) \ , \forall \mathbf{x} \mid \mathbf{x} \in \mathcal{F}, \mathbf{x} \notin \mathcal{X}$$

III. Constrained Niching Algorithms

To investigate multimodal optimization of constrained functions, a number of conventional niching algorithms that use DE and have been shown to perform well at unconstrained multimodal optimization, are combined with a common constraint handling method also often applied to DE to produce constrained niching techniques. In this section, these are fully outlined and further developed via a parallel decomposition to permit the use of expensive objective function evaluations. First, the canonical differential evolution algorithm is presented, on which the constrained niching methods are built.

III.A. Canonical Differential Evolution

The idea of DE was first presented by Storn and Price [7, 46] and has since been developed into a widely-used global search algorithm. A full review on the development and applications of DE is outside the scope of this paper, however, in-depth reviews have been presented by Das and Suganthan [47], Neri and Tirronen [48] and more recently by Das *et al.* [49].

DE uses a number of individuals who evolve through the iterations of the optimization. The total population is composed of N individuals, where the n-th individual is represented by a target vector that details an individual's position in the design space. The nomenclature used in this paper to represent the target vector of the n-th individual at the t-th iteration of the optimization, is given by:

$$\mathbf{x}_n(t) = [x_n^1(t), x_n^2(t), \ldots, x_n^D(t)]^T \tag{2}$$

DE follows five steps to advance the optimization algorithm, which are given as:

1. *Initialisation*:

   Within the design space $\mathcal{S}$, the initial target vectors of the N individuals are generated. This is commonly done randomly such that the target vector at the 0-th iteration (i.e. the initial location of the individual) in the d-th dimension ($d \in (1, D)$) is given as:

   $$x_n^d(0) = L^d + \text{rand}(0,1)(U^d - L^d) \tag{3}$$

   where $\text{rand}(0,1)$ is a uniformly distributed random number on the interval 0 to 1.

2. *Mutation*:

   For each individual within the population, a donor vector is generated using scaled differences between other individuals within the population. A number of mutation strategies have been proposed but the DE/rand/1 strategy is used throughout this paper unless otherwise specified. The n-th donor vector, $\mathbf{v}_n$, using a DE/rand/1 mutation strategy, is given by:

   $$\mathbf{v}_n = \mathbf{x}_{r_1}(t) + F(\mathbf{x}_{r_2}(t) - \mathbf{x}_{r_3}(t)) \tag{4}$$

   where F is the difference-vector scaling factor, and r_1, r_2 and r_3 are uniformly distributed random integers on the interval $(1, N)$ such that $r_1 \neq r_2 \neq r_3 \neq n$.

3. <u>*Crossover*</u>:

   To enhance diversity, crossover of the individual elements of the target vector and the donor vector is employed to produce a trial vector. Binomial crossover is used throughout this paper which produces the n-th trial vector in the d-th dimension, u_n^d, by the following equation:

$$u_n^d = \begin{cases} v_n^d & \text{if } \mathrm{rand}(0,1) \leq CR \text{ or } r_n = d \\ x_n^d(t) & \text{otherwise} \end{cases} \tag{5}$$

   where r_n is a uniformly distributed random integer on the interval $(1, D)$ and CR is the crossover probability. Hence, elements of the trial vector are accepted from the donor vector at a probability of CR, and at least one component of the donor vector is accepted.

4. <u>*Selection*</u>:

   The trial vector is tested and the n-th target vector at the next iteration is generated by the following relationship:

$$\mathbf{x}_n(t+1) = \begin{cases} \mathbf{u}_n & \text{if } f(\mathbf{u}_n) \leq f(\mathbf{x}_n(t)) \\ \mathbf{x}_n(t) & \text{otherwise} \end{cases} \tag{6}$$

   Hence, the trial vector is selected if it is in a 'better' location in the design space, otherwise the target vector is conserved.

5. <u>*Stopping conditions*</u>:

   Once all N target vectors have been updated, if the stopping conditions have been reached (which throughout this paper is whether the maximum allowed number of function evaluations have been performed), the algorithm exits. If not, then the processes of steps 2-5 are repeated.

The canonical DE algorithm used here uses rand/1 mutation and binomial crossover, so keeping with the standard DE nomenclature, this is the DE/rand/1/bin algorithm. Figure 1 demonstrates, first, the mutation mechanism within DE, which is illustrated in red, where a scaled difference vector between two randomly sampled individuals is applied to a further random individual. The binomial crossover mechanism is then illustrated in blue where elements of $\mathbf{v}_n$ are selected to produce a trial vector. In this example, $\widehat{\mathbf{u}_n}$ and $\widetilde{\mathbf{u}_n}$ are the two possible trial vectors that could be generated.

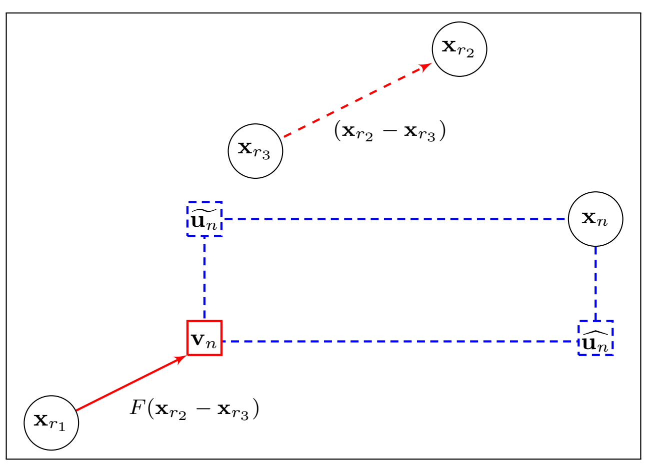


Figure 1: Illustration of mutation (red) and crossover (blue) in differential evolution

From the algorithm above, it is clear that there is no direct way to handle constraints within DE so *ad hoc* methods have to be employed (see the book of Datta and Deb for full discussions on these [50]). One common approach is to employ so-called feasibility rules [15], where selection is performed based on the feasibility of individuals. For example, given two feasible individuals, the one with the best fitness wins. On the other hand, if one is feasible and one infeasible, the feasible one always wins. While, if both are infeasible, then the one that has the smallest violation of constraints is the winner. This can be written as a domination operator, where, given two locations $\mathbf{x}_a$ and $\mathbf{x}_b$, $\mathbf{x}_b$ dominates $\mathbf{x}_a$ based on the following:

$$\mathbf{x}_a \prec \mathbf{x}_b \Leftrightarrow \begin{cases} f(\mathbf{x}_b) < f(\mathbf{x}_a) & \text{and} \quad \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) = 0 \\ \phi(\mathbf{x}_b) = 0 & \text{and} \quad \phi(\mathbf{x}_a) > 0 \\ \phi(\mathbf{x}_b) < \phi(\mathbf{x}_a) & \text{and} \quad \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) > 0 \end{cases} \tag{7}$$

where ϕ is the constraint violation given by:

$$\phi(\mathbf{x}) = \sum_{j=1}^{p} \max[0, g_j(\mathbf{x})] + \sum_{j=1+p}^{m} |h_j(\mathbf{x})|$$

In DE, these feasibility rules are commonly used in the selection step to determine whether the trial vector should replace the target vector. Hence, rewriting equation 6 for constrained optimization leads to:

$$\mathbf{x}_n(t+1) = \begin{cases} \mathbf{u}_n & \text{if } \mathbf{x}_n(t) \prec \mathbf{u}_n \\ \mathbf{x}_n(t) & \text{otherwise} \end{cases} \tag{8}$$

The constraint handling for DE has now been introduced. Recent work by the authors has considered performing multimodal optimization in constrained spaces and a number of algorithms have been developed at the University of Bristol. A number are developed further here for performing multimodal aerodynamic optimization. These are:

- fDE: feasible DE [a]
- fNRAND1: feasible DE using nrand1 [27] mutation
- fINRAND1: feasible DE using inrand1/r [28] (nearest neighbour with ring network) mutation
- fCDE: feasible form of crowding DE [20]
- fNCDE: feasible form of neighbourhood-based CDE [52]

When performing aerodynamic optimization, one of the characteristics is an expensive objective function evaluation, which is a CFD evaluation. In a population-based optimization algorithm, $O(10^2)$ objective function evaluations may be required each iteration so clearly a parallel decomposition of the search population becomes necessary. However, the original form of DE, as presented by Storn and Price [7], formulates DE in a sequential manner, where a single loop (over each individual) contains the mutation, crossover, evaluation and selection stage. This form of DE is shown schematically in figure 2. On the other hand, a parallel decomposition requires that the objective evaluation be in its own loop, such that this can be assigned to each processor. This is shown in figure 3. In the following sections, the various constrained niching methods are introduced and developed using parallel decomposition in the format seen in figure 3.

[a] First presented by Mezura-Montes *et al.* [51]

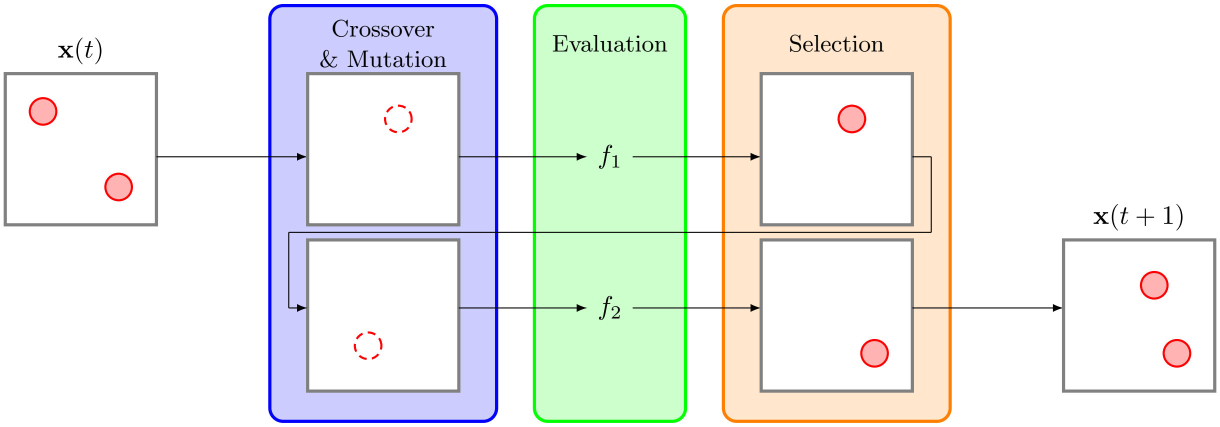


Figure 2: Sequential form of DE (example using two individuals)

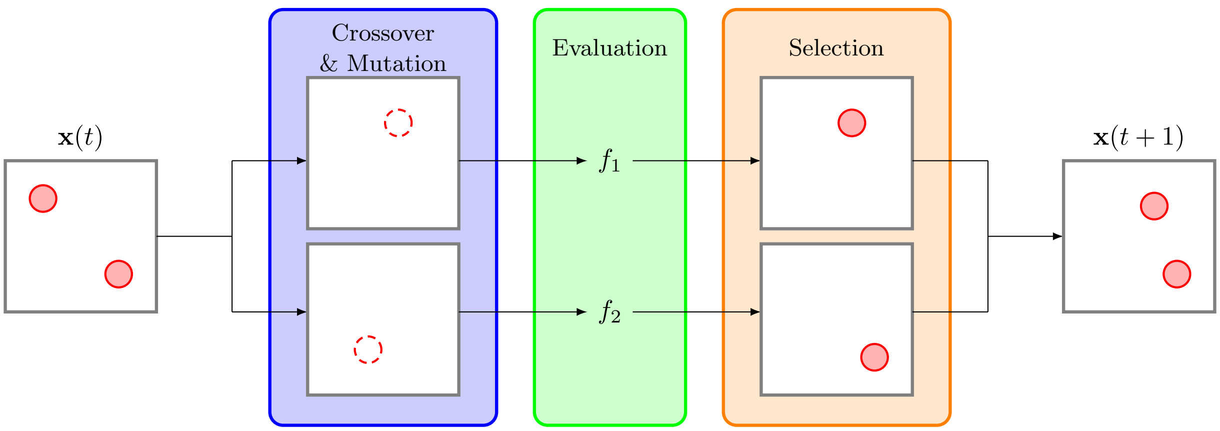


Figure 3: Parallel form of DE (example using two individuals)

III.B. fDE

fDE uses the canonical DE algorithm with equation 8 used for the selection stage. The rand/1/bin strategy is used. The development of a parallel form of the algorithm requires that the objective and constraint evaluation be evaluated for one individual on its own process. This information is then sent back to the master process which then performs the update of the locations of each individual. The overall algorithm in serial and parallel is outlined as algorithm 1.

Algorithm 1 fDE algorithm in serial (left) and parallel (right)

Serial (left):

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \to N$ **do**
    Perform rand/1 mutation: equation 4
    Perform binomial crossover: equation 5
    Objective of trial vector
    Update n-th target vector: equation 8
  end for
end while

Parallel (right):

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \to N$ **do**
    Perform rand/1 mutation: equation 4
    Perform binomial crossover: equation 5
  end for
  for $n = 1 \to N$ **do**
    if $n = procid$ **then**
      Objective of n-th trial vector
    end if
  end for
  for $n = 1 \to N$ **do**
    Update n-th target vector: equation 8
  end for
end while

III.C. fNRAND1

The fNRAND1 algorithm uses the target vector of the n-th individual's nearest neighbour, $\mathbf{x}_{NN_n}$, as the base vector against which to provide the difference vector in the mutation stage. The mutation is given as:

$$\mathbf{v}_n = \mathbf{x}_{NN_n}(t) + F(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{9}$$

where:

$$NN_n = \underset{i \in (1,N), i \neq n}{\arg\min} \; \|\mathbf{x}_n - \mathbf{x}_i\|_2$$

The selection stage uses equation 8 for feasibility. The overall algorithm is outlined in algorithm 2, again in serial and parallel.

Algorithm 2 fNRAND1 algorithm in serial (left) and parallel (right)

Serial:

```
Randomly initialise individuals, calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest neighbour to x_n
        Perform nrand/1 mutation: equation 9
        Perform binomial crossover: equation 5
        Objective of trial vector
        Update n-th target vector: equation 8
    end for
end while
```

Parallel:

```
Randomly initialise individuals, calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest neighbour to x_n
        Perform nrand/1 mutation: equation 9
        Perform binomial crossover: equation 5
    end for
    for n = 1 → N do
        if n = procid then
            Objective of n-th trial vector
        end if
    end for
    for n = 1 → N do
        Update n-th target vector: equation 8
    end for
end while
```

III.D. fINRAND1

The fINRAND1 algorithm uses the target vector of n-th individual's nearest neighbour within its local neighbourhood, $\mathbf{x}_{INN_n}$, as the base vector against which to provide the difference vector in the mutation stage. In this work, an index-based ring neighbourhood is used hence this reduces computational complexity against fNRAND1. The mutation is given as:

$$\mathbf{v}_n = \mathbf{x}_{INN_n}(t) + F(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{10}$$

where:

$$INN_n = \begin{cases} \underset{i \in \{N,2\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{if } n = 1 \\ \underset{i \in \{N-1,1\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{if } n = N \\ \underset{i \in \{n-1,n+1\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{otherwise} \end{cases}$$

The selection stage uses equation 8 for feasibility. The overall algorithm is outlined in algorithm 3.

Algorithm 3 fINRAND1 algorithm in serial (left) and parallel (right)

Serial:

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \rightarrow N$ **do**
    Find the nearest neighbour to $\mathbf{x}_n$ in ring
    Perform inrand/1 mutation: equation 10
    Perform binomial crossover: equation 5
    Objective of trial vector
    Update n-th target vector: equation 8
  end for
end while

Parallel:

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \rightarrow N$ **do**
    Find the nearest neighbour to $\mathbf{x}_n$ in ring
    Perform inrand/1 mutation: equation 10
    Perform binomial crossover: equation 5
  end for
  for $n = 1 \rightarrow N$ **do**
    if $n = procid$ **then**
      Objective of n-th trial vector
    end if
  end for
  for $n = 1 \rightarrow N$ **do**
    Update n-th target vector: equation 8
  end for
end while

III.E. fCDE

The fCDE algorithm uses the normal CDE algorithm but with the feasible selection method. This requires creating a trial vector using the rand/1/bin strategy, and once this is found, the closest individual to the trial vector, $\mathbf{x}_{u_n}$, needs to be found. Once this is found, this closest individual is replaced by the trial vector if the trial vector is better, determined using the feasibility rules:

$$\mathbf{x}_{u_n}(t+1) = \begin{cases} \mathbf{u}_n & \text{if } \mathbf{x}_{u_n}(t) \prec \mathbf{u}_n \\ \mathbf{x}_{u_n}(t) & \text{otherwise} \end{cases} \tag{11}$$

The fCDE algorithm is outlined in algorithm 4.

Algorithm 4 fCDE algorithm in serial (left) and parallel (right)

Serial:

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \rightarrow N$ **do**
    Perform rand/1 mutation: equation 4
    Perform binomial crossover: equation 5
    Objective of trial vector
    Find the closest individual to $\mathbf{u}_n$
    Update closest individual: equation 11
  end for
end while

Parallel:

Randomly initialise individuals, calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \rightarrow N$ **do**
    Perform rand/1 mutation: equation 4
    Perform binomial crossover: equation 5
  end for
  for $n = 1 \rightarrow N$ **do**
    if $n = procid$ **then**
      Objective of n-th trial vector
    end if
  end for
  for $n = 1 \rightarrow N$ **do**
    Find the closest individual to $\mathbf{u}_n$
    Update closest individual: equation 11
  end for
end while

III.F. fNCDE

The feasible neighbourhood algorithm of crowding DE, fNCDE, generates a trial vector from a neighbourhood that is made up from the m nearest individuals to the n-th individual. Hence, when performing rand/1 mutation (equation 4), r_1, r_2 and r_3 are uniformly distributed random integers that come from the set of integers that represent the m nearest neighbours. Once a trial vector is found, updating occurs on the whole population according to normal crowding DE, where the nearest neighbour to the trial vector is used for comparison. The algorithm is given in algorithm 5.

Algorithm 5 fNCDE algorithm in serial (left) and parallel (right)

```
Randomly initialise individuals, calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest m individuals to x_n
        Perform rand/1 mutation using the nearest m
        Perform binomial crossover: equation 5
        Objective of trial vector
        Find the closest individual to u_n
        Update closest individual: equation 11
    end for
end while
```

```
Randomly initialise individuals, calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest m individuals to x_n
        Perform rand/1 mutation using the nearest m
        Perform binomial crossover: equation 5
    end for
    for n = 1 → N do
        if n = procid then
            Objective of n-th trial vector
        end if
    end for
    for n = 1 → N do
        Find the closest individual to u_n
        Update closest individual: equation 11
    end for
end while
```

IV. Analytical Optimization Example

To demonstrate the effectiveness of the state-of-the-art parallel niching methods, optimizations using the algorithms above on an analytical test problem are considered in this section.

The test problem has two design variables, x_1 and x_2, and uses the Himmelblau function as the objective function with the addition of four constraints. The objective function to be minimised is:

$$f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 1$$

subject to:

$$\begin{aligned}
g_1(\mathbf{x}) &= \zeta_1 x_1 + \zeta_2 x_1 - \zeta_1\zeta_2 + \eta_1 x_2 + \eta_2 x_2 - \eta_1\eta_2 - x_1^2 - x_2^2 \leq 0 \\
g_2(\mathbf{x}) &= \zeta_2 x_1 + \zeta_3 x_1 - \zeta_2\zeta_3 + \eta_2 x_2 + \eta_3 x_2 - \eta_2\eta_3 - x_1^2 - x_2^2 \leq 0 \\
g_3(\mathbf{x}) &= \zeta_3 x_1 + \zeta_4 x_1 - \zeta_3\zeta_4 + \eta_3 x_2 + \eta_4 x_2 - \eta_3\eta_4 - x_1^2 - x_2^2 \leq 0 \\
g_4(\mathbf{x}) &= \zeta_4 x_1 + \zeta_1 x_1 - \zeta_4\zeta_1 + \eta_4 x_2 + \eta_1 x_2 - \eta_4\eta_1 - x_1^2 - x_2^2 \leq 0
\end{aligned}$$

where $\boldsymbol{\zeta} = [3.0, -2.805, -3.779, 3.584]$ and $\boldsymbol{\eta} = [2.0, 3.131, -3.283, -1.848]$. The design space bounds are $-6 \leq x_d \leq 6$ $(d = 1, 2)$. The problem has four global optima:

$$\begin{aligned}
\mathbf{x}^{*1} &= [3, 2]^T \\
\mathbf{x}^{*2} &= [-2.805118, 3.131312]^T \\
\mathbf{x}^{*3} &= [-3.779310, -3.283186]^T \\
\mathbf{x}^{*4} &= [3.584428, -1.848126]^T
\end{aligned}$$

which each have an optimal objective value of 1.0. At any one optima, two constraints are active. Figure 4 shows the design space of the problem, where the lines show the active constraint boundaries (the solution is feasible outside of these boundaries) and the symbols show the four optima.

IV.A. Run Details

The number of runs on each function, NR, is 50 for each of the algorithms to ensure sufficient sampling on which statistical analysis can be performed. The remainder of the optimization parameters are given in table 1. The population size was chosen to give a good balance between cost and performance. The initial location for each was random within the design space bounds, while bound constraints were handled by randomly

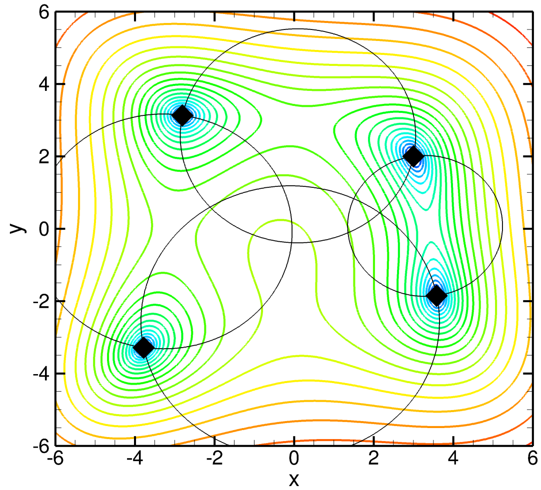


Figure 4: Analytical test problem design space and active constraint lines (symbols show optima)

reinitialising an individual if it exited the design space bounds. When performing the runs, no knowledge of the functions is assumed by the algorithms, so they are seen as entirely black-box.

Table 1: Optimization parameters

Parameter	Value
N	200
FEs_{max}	400,000
F	0.9
CR	0.1
m	10

IV.B. Results

To determine the performance of the state-of-the-art constrained niching methods, two primary performance metrics are considered. These are the peak accuracy, PA, and the peak ratio, PR. Peak accuracy is a measure to determine how closely each method identifies the peaks within the search space. It is the average over the total number of peaks of the nearest search agent (in the objective space) to each peak:

$$PA = \frac{1}{N_g} \sum_{1}^{N_g} \min_{n \in (1,N)} |f(\mathbf{x}^*) - f(\mathbf{x}_n)| \tag{12}$$

where N_g is the number of global optima (four in this case). Peak ratio is the number of optima found as a fraction of the total number of optima:

$$PR = \frac{N_{found}}{N_g} \tag{13}$$

where an error is used to determine how N_{found} is calculated.

Table 2 gives the average PA over the 50 runs and the average PR over the 50 runs at five different tolerance levels. Clearly, the fNCDE and fNRAND1 algorithms converge to the best number of optima, and both have very low average PA values. Interestingly, fDE converges very far down in the objective space, however tends to only converge onto one optima. This is something that is also shown in figure 5 (value of PR through optimization history) and 6 (location of search agents), where fDE starts converging onto all four global optima, however then loses diversity and results in one overall converged population.

Table 2: Average peak accuracy and peak ratio (at different error levels) of methods

		PR				
Algorithm	PA	1E-1	1E-2	1E-3	1E-4	1E-5
fCDE	7.67×10^{-5}	1.000	1.000	1.000	0.765	0.100
fDE	1.58×10^{-20}	0.250	0.250	0.250	0.250	0.250
fINRAND1	7.31×10^{-6}	1.000	1.000	1.000	0.995	0.855
fNCDE	5.51×10^{-14}	1.000	1.000	1.000	1.000	1.000
fNRAND1	5.55×10^{-18}	1.000	1.000	1.000	1.000	1.000

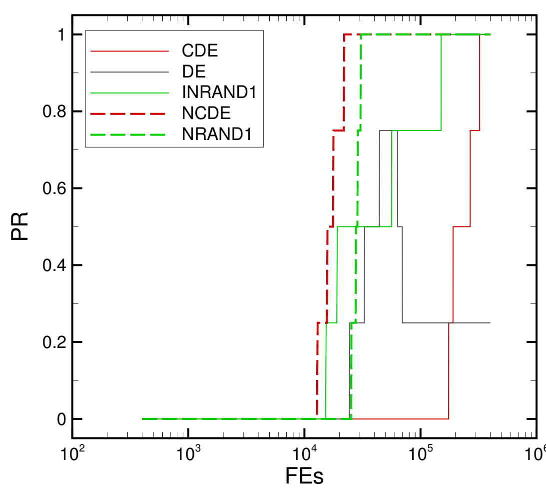
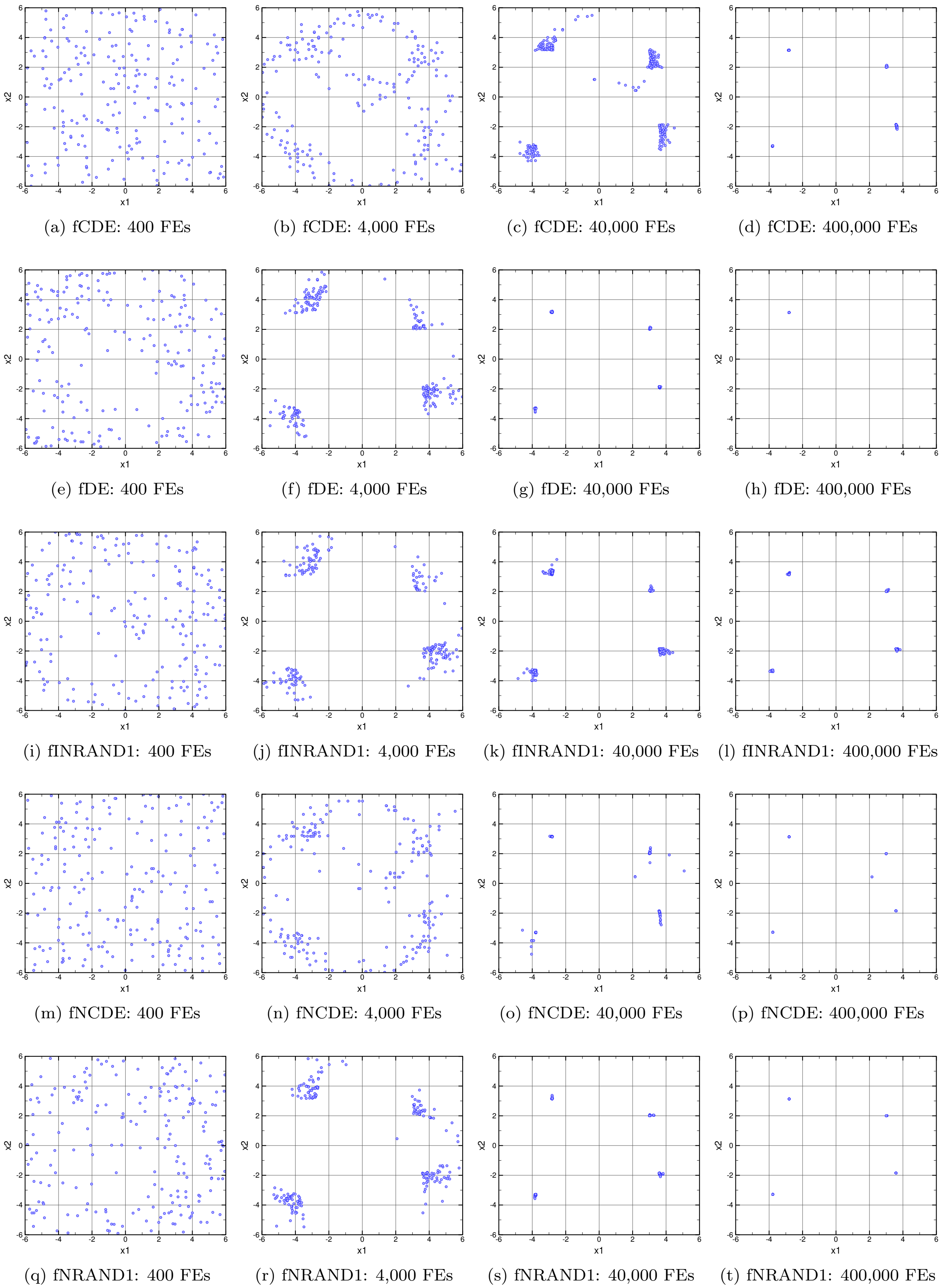


Figure 5: Convergence of PR at $\epsilon = 10^{-3}$



(a) fCDE: 400 FEs (b) fCDE: 4,000 FEs (c) fCDE: 40,000 FEs (d) fCDE: 400,000 FEs

(e) fDE: 400 FEs (f) fDE: 4,000 FEs (g) fDE: 40,000 FEs (h) fDE: 400,000 FEs

(i) fINRAND1: 400 FEs (j) fINRAND1: 4,000 FEs (k) fINRAND1: 40,000 FEs (l) fINRAND1: 400,000 FEs

(m) fNCDE: 400 FEs (n) fNCDE: 4,000 FEs (o) fNCDE: 40,000 FEs (p) fNCDE: 400,000 FEs

(q) fNRAND1: 400 FEs (r) fNRAND1: 4,000 FEs (s) fNRAND1: 40,000 FEs (t) fNRAND1: 400,000 FEs

Figure 6: Convergence of individuals on function `F11`

To ensure that the comparisons of the five algorithms is statistically significant, Wilcoxon rank-sum tests[53] are performed. The rank-sum test is used to test the null hypothesis that "*the median of the peak ratios of algorithms A and B are equal*" where the sample-set is all of the peak ratios used to produce the data in table 2. The confidence level of the test is 99%. Both right-tailed and left-tailed p-values are calculated to test the alternate hypotheses of whether the median of A is greater than the median of B (if this is the case, then A is said to have won) and whether the median of A is less that the median of B (then B is said to have won), respectively. If the null hypothesis is accepted then there is no difference at the confidence level.

Figure 7 gives the results of all the comparisons between all five algorithms. In this figure, if a box is green, then the algorithm in the row wins against the algorithm in the column, while if it is red then the row loses against the column and if the box is white then there is no statistical difference. From these results, it is relatively simple to determine the ranking of the five algorithms, with fNCDE and fNRAND1 being the best, followed by fINRAND1, fCDE and fDE. Hence, there is no statistical difference in the peak ratio performance between fNCDE and fNRAND1, however, the PA results do show that fNRAND1 is able to converge to a lower tolerance than fNCDE.

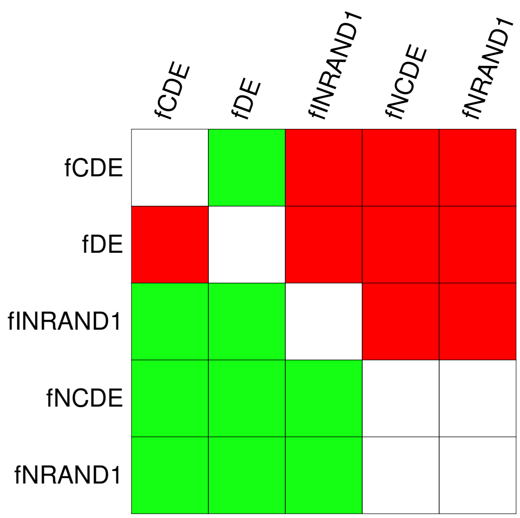


Figure 7: Results of rank-sum tests on peak ratios

V. Multimodal Aerodynamic Wing Optimization Problem

It has been shown above that reformulating niching algorithms via parallel decomposition to allow the use of an expensive objective function evaluation is a suitable approach. The motivation for such development is such that these advanced algorithms can be applied to allow the identification of any optimality in aerodynamic optimization design spaces. In this section, the fNRAND1 niching algorithm is run on an example aerodynamic wing optimization case that exhibits multimodality, and the performance is compared to running a gradient-based algorithm from multiple start locations.

V.A. Problem Definition

The problem is based on the AIAA ADODG Case 6, which involves drag minimization of a rectangular wing subject to aerodynamic constraints on lift, C_L, and root bending moment, C_{M_x}, and geometric constraints on wing area, S, internal volume, V, twist, γ, local chord, $c(y)$, local thickness, $t(y)$, sweep (local x deformation at the quarter-chord), $\Delta x_{qc}(y)$, semi-span, s, dihedral (local z deformation at the quarter chord), $\Delta z_{qc}(y)$, and angle of attack, AoA. The problem is given by:

$$\begin{aligned}
\underset{\boldsymbol{\alpha}\in\Re^D}{\text{minimise}} \quad & \qquad C_D \\
\text{subject to} \quad & C_L = 0.2625 \\
& C_{M_x} \leq 0.1069 \\
& S = S(\text{initial}) \\
& V \geq V(\text{initial}) \\
& -3.12^\circ \leq \gamma \leq 3.12^\circ \\
& 0.45 \leq c(y) \leq 1.55 \quad \forall y \in [0, s] \\
& 0.06 \leq t(y) \leq 0.18 \quad \forall y \in [0, s] \\
& -1 \leq \Delta x_{qc}(y) \leq 1 \quad \forall y \in [0, s] \\
& 2.46 \leq s \leq 3.67 \\
& -0.45 \leq \Delta z_{qc}(y) \leq 0.45 \quad \forall y \in [0, s] \\
& -3.0^\circ \leq AoA \leq 6.0^\circ
\end{aligned} \tag{14}$$

As previously covered in Poole *et al.* [37], the root bending moment constraint of this problem inhibits the conventional optimal elliptic loading result. Instead, the optimal loading of this result is given by the solution of Jones [54] (which is a specific solution of a result later proved by Klein and Viswanathan [55]):

$$\Gamma(\eta) = \Gamma_0 \left[(3 - 2\epsilon)\sqrt{1 - \eta^2} + 6(\epsilon - 1)\eta^2 \cosh^{-1}\left(\frac{1}{|\eta|}\right) \right] \tag{15}$$

where:

$$\epsilon = \frac{3\pi}{2}\frac{C_{M_x}}{C_L}$$

Γ_0 is a scaling factor and $\eta = y/s$ is the non-dimensional spanwise location. The theoretical minimum induced drag can be calculated as:

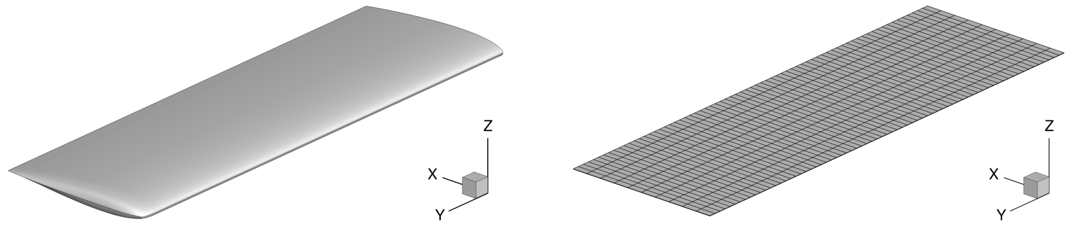
$$C_{D_i} = \frac{C_L^2}{\pi AR}(1 + \delta)$$

where $\delta = 8(\epsilon - 1)^2$, hence this varies with aspect ratio. Poole *et al.* [37] showed that the theoretical minimum for this problem occurs at the maximum allowable span, and is 24.7 counts.

However, while Case 6 is interesting, it permits a substantial degree of geometric flexibility, hence it is useful to consider a slightly simpler problem. The problem considered here is the drag minimization of the Case 6 wing subject to variations in chord, local sweep and span only. The half-wing (shown in figure 8) is composed of a rectangular NACA0012 section of span 3.0 and a rounded wing-cap of width 0.06, resulting in the overall semi-span of 3.06. The problem specifies that the wing is in compressible, inviscid flow at $M_\infty = 0.5$, however, for this problem, a vortex-lattice solver is used to determine the aerodynamic forces and moments. Hence, the flow is incompressible and inviscid with no compressibility scaling performed. The wing is trimmed to an incompressible target $C_L = 0.2625$.

A vortex lattice solver (based on that defined in Katz and Plotkin [56]) is used to determine the aerodynamic forces and moments. The NACA0012 half-wing is approximated as a flat surface on the chord plane of the wing, and this is discretised into a grid of 10 chordwise panels by 70 spanwise panels (see figure 8). This half mesh is then mirrored in the x-z plane for the VLM solver.

Deformation of the aerodynamic surface and mesh is achieved via a lattice of control points that use radial basis function (RBF) interpolation to link changes in control point positions to changes in the surface and mesh. The control point set-up used is shown in figure 9. Hence, control point deformations act of design variables. Local (sectional-based) changes in chord and sweep occur at five evenly-spaced spanwise stations. A global variations in pitch results in an eleven design variable problem. A global span variable also exists, however this is calculated to maintain a fixed planform area after planform deformations have occurred.



(a) Surface (b) VLM mesh

Figure 8: Rectangular NACA0012 wing and VLM discretisation

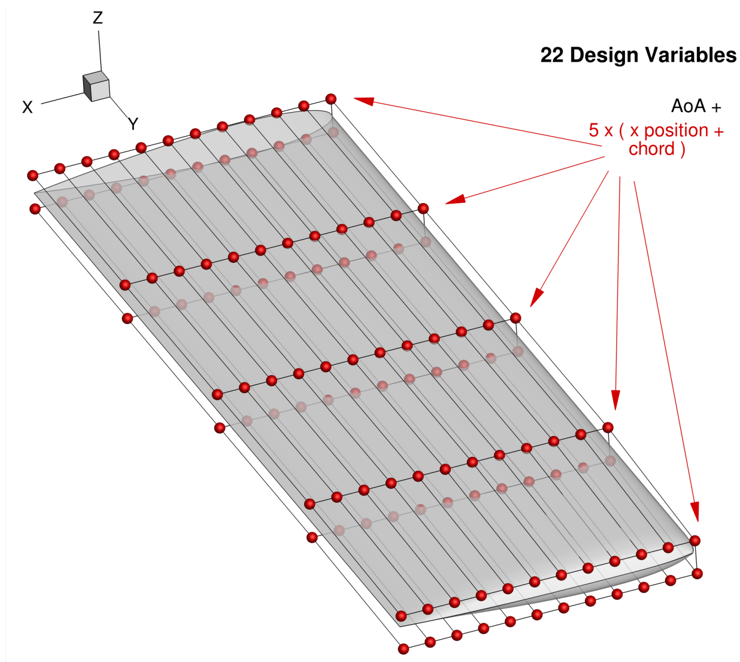


Figure 9: Wing embedded in control point cage

The design variable specification is as follows:

- x_1 - angle of attack
- x_2 to x_6 - chord changes at spanwise stations
- x_7 to x_{11} - sweep changes at spanwise stations

The optimization problem being solved by the optimizer is:

$$
\begin{array}{lrl}
\underset{\mathbf{x}\in\Re^D}{\text{minimise}} & C_D & \\
\text{subject to} & 0.2625 - C_L \leq 0 & \\
 & C_{M_x} - 0.1069 \leq 0 & \\
 & 2.46 - s \leq 0 & \\
 & s - 3.67 \leq 0 & \\
 & & \\
 & 3.0 \leq x_1 \leq 6.0 & \\
 & 0.45 \leq x_i \leq 1.55 & \forall i \in [2,6] \\
 & -1 \leq x_i \leq 1 & \forall i \in [7,11]
\end{array}
\tag{16}
$$

noting that the wing area remains fixed throughout the optimization and that a flat plate is used for a VLM so the internal volume constraint is void.

V.B. Multi-Start Gradient-Based Results

To demonstrate multimodality of this problem, a number of independent runs of a local optimizer are performed where each run represents a different starting location in the design space. The gradient-based optimization algorithm is the feasible sequential quadratic programming (FSQP) algorithm as implemented in version 3.7 [57]. FSQP is based on the sequential quadratic programming (SQP) approach, but modified to improve convergence by combining a search along an arc [58] with a non-monotone procedure for that search [59]. The FSQP algorithm is fully described and analysed in [60, 61]. Gradients are evaluated using a second-order central difference scheme, so two objective evaluations are required per design variable gradient. 30 independent runs of the optimizer are performed where a randomised Latin hypercube sampling procedure is used to determine the starting location of each of the 30 runs. A further 'datum' run is also performed where the starting location is the baseline wing shape without any initial deformations applied to it.

Table 3 gives the initial results and the average of all of the optimum results. All optima converged to 24.7 counts (this is very close to the theoretically obtainable minimum from theory) and were within 0.01 counts of each other. Figure 10 shows an example convergence of the feasibility and optimality from one of the 30 runs (this is a typical example), showing that the solution begins infeasible so a feasible point has to first be found. Once this occurs, the optimizer can work on improving the objective function.

Table 3: Gradient-based wing optimization results (C_D in counts)

	C_L	C_D	C_{M_x}	s	$S/2$	AoA	$\Delta J(\%)$
Baseline	0.263	36.8	0.118	3.00	3.00	3.47°	-
Mean result	0.263	24.7	0.107	3.67	3.00	3.10°	**-32.9%**

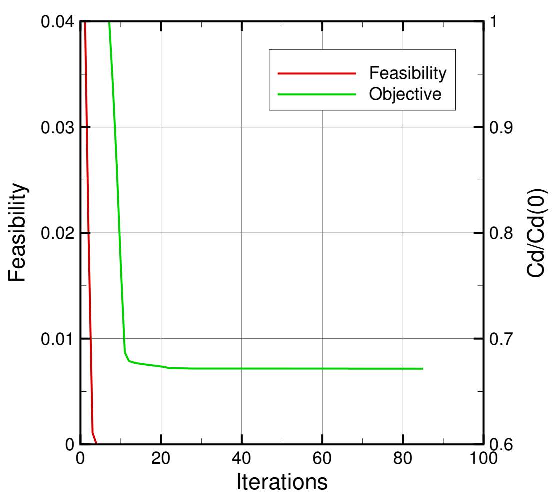


Figure 10: Objective and feasibility convergence from one run of gradient-based algorithm

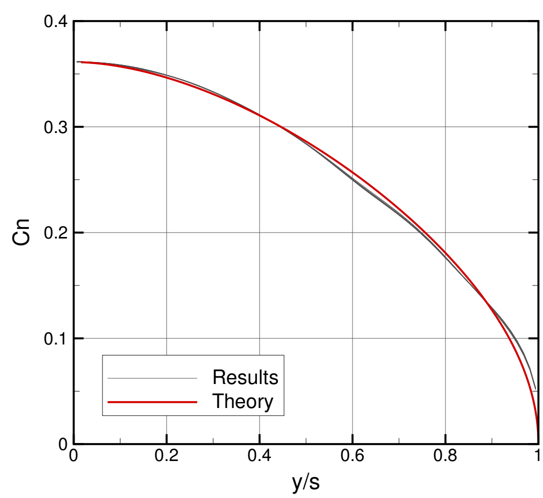
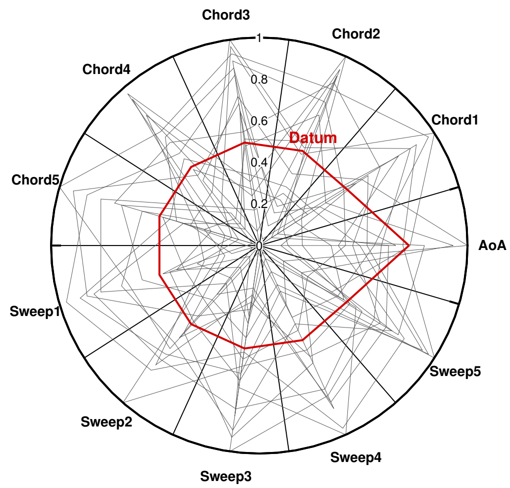
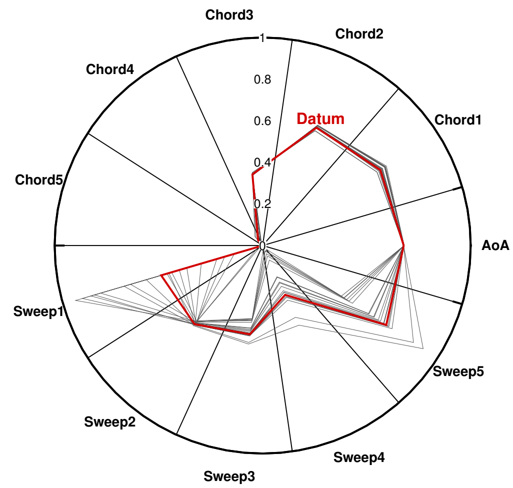


Figure 11: Wind loading of different optima from gradient-based algorithm

While the optimum results of all runs are near identical, looking at the final design variable values (given in figure 12) it can be seen that the optimum results all have differing design variable values. There is clear multimodality present but this appears to come from the sweep, rather than the chord. The final spanwise loadings are given in figure 11 while the wing shapes are given in figure 13. All results are almost identical in terms of loading, and are very close to the theoretical result. However, the surfaces appear to group into three different shapes, with two results showing large forward sweep, the majority of results (including the datum result) showing moderate forward sweep, then another group with less forward sweep.



(a) Initial locations



(b) Optimum locations

Figure 12: Star maps showing initial and optimum values of design variables from gradient-based algorithm

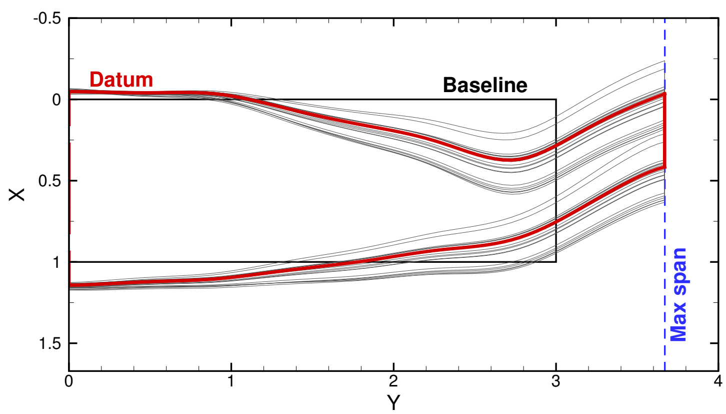
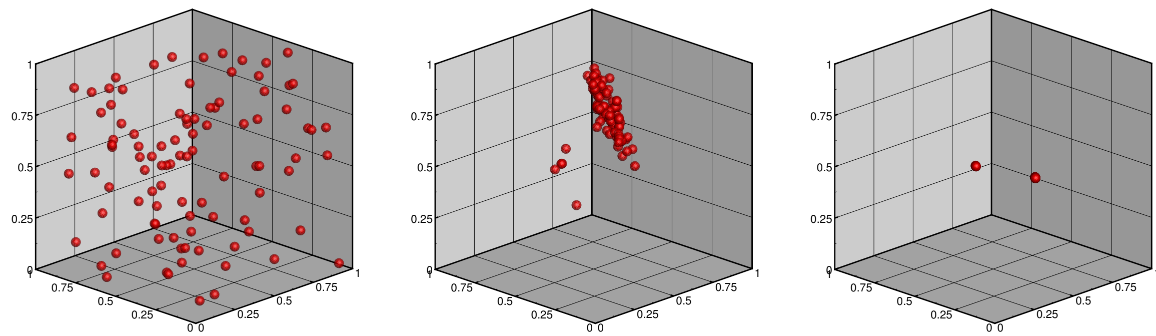


Figure 13: Surface shapes of different optima from gradient-based algorithm

V.C. Niching Results

The results of running gradient-based optimization from different starting locations demonstrates that this problem is multimodal. However, the shapes of the different optima are remarkably similar, and it appears that the optimum is something akin to a valley, containing lots of very small, local minima. Despite this, it is interesting to consider the performance of niching algorithms on this problem. The niching algorithm tested here is fNRAND1, which is run with 96 individuals in the population.

The location of the agents in three of the design variables at different points during the niching optimization process are shown in figure 14. It is clear that the population is converging onto two individually identifiable optima. The design variable values of these two optima are shown in the star map in figure 15 and these are compared to the optima found in the gradient-based results. Interestingly, one of the optima has similar trends to the results found in the gradient-based optimizations, however, the other optima is substantially different. From table 4 the optimum that is similar to the gradient-based results has similar performance, however the second optimum is a local optimum that is slightly worse than the first. The lift, root bending moment and upper span constraints are active in both, however from figure 16, there are very stark differences in the shapes of the two optima. It is positive that from these initial investigations, niching finds minima that were not found from an initial study of multi-start gradient-based optimization.



(a) $t = 0$ (b) $t = 0.25T$ (c) $t = T$

Figure 14: Population state of three example design variables through niching process

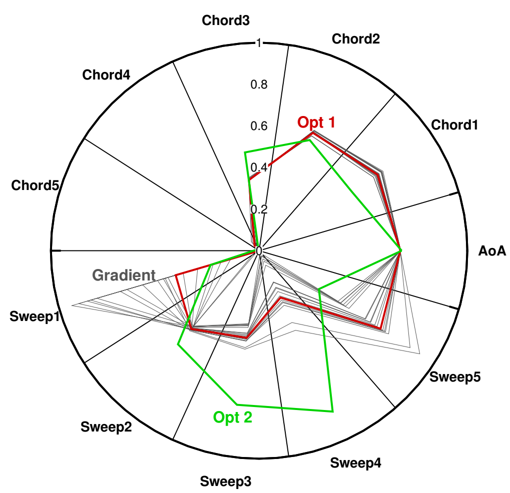


Figure 15: Star map showing optimum values of design variables from niching algorithm

Table 4: Niching using fNRAND1 wing optimization results (C_D in counts)

	C_L	C_D	C_{M_x}	s	$S/2$	AoA	$\Delta J(\%)$
Baseline	0.263	36.8	0.118	3.00	3.00	3.47°	-
Gradient result	0.263	24.7	0.107	3.67	3.00	3.10°	**-32.9%**
Optimum 1	0.263	24.7	0.107	3.67	3.00	3.11°	**-32.9%**
Optimum 2	0.263	25.1	0.107	3.67	3.00	3.16°	**-31.8%**

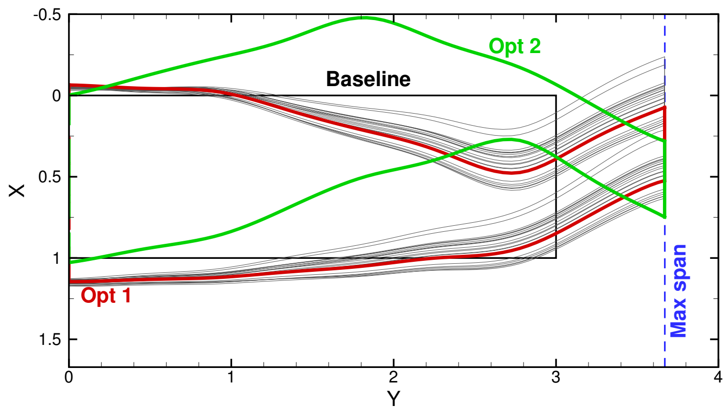


Figure 16: Surface shapes of different optima from niching algorithm

VI. Conclusions and Ongoing Work

The work in this paper has considered locating multiple optima in aerodynamic optimization design spaces. To perform this, state-of-the-art constrained niching methods that have previously been shown to perform well when locating multiple optima in analytical problems have been employed. These method use a population of search agents who cluster together at different locations within the search space to converge onto multiple optima. However, the use of such techniques for locating multiple optima in aerodynamic optimization spaces becomes problematic due to the high cost of the objective function evaluation. As such, a parallel decomposition of each of the methods has been developed to allow each individual to be evaluated in parallel, limiting the cost of each optimizer iteration to a single objective function evaluation. The fNRAND1 and fNCDE niching algorithms were shown to perform similarly on an example analytical problem, and these were better (at 99% confidence) than the other algorithms tested.

A wing planform optimization case has been tested with a vortex lattice solver used for the aerodynamic performance evaluation. Multi-start gradient-based optimization was used to determine multimodality of the case, but that this multimodality followed similar trends in design variables with small variations in sweep providing the multimodality. The fNRAND1 niching algorithm was then employed on this case and two individual minima were found; one which follows similar trends to the gradient-based and one which is substantially different. The first optimum found had better performance than the second optima, indicating that this problem has a degree of local multimodality. It is positive to note that the niching algorithm was able to locate multiple optima on an aerodynamic optimization case, indicating the potential of these algorithms for examples beyond simply analytical problems.

Future and ongoing work includes considering reducing the cost associated with niching algorithms as well as applying these methods to further aerodynamic optimization problems.

References

[1] Kenway, G. K. W. and Martins, J. R. R. A., "Multi-point High-fidelity Aerostructural Optimization of a Transport Aircraft Configuration," *Journal of Aircraft*, Vol. 51, No. 1, 2014, pp. 144–160. doi:10.2514/1.C032150.

[2] Mishra, A., Mavriplis, D., and Sitaraman, J., "Time-Dependent Aeroelastic Adjoint-Based Aerodynamic Shape Optimization of Helicopter Rotors in Forward Flight," *AIAA Journal*, Vol. 54, No. 12, 2016, pp. 3813–3827. doi:10.2514/1.J054962.

[3] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016, pp. 828–846. doi:10.2514/1.J053813.

[4] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer," *Computers & Fluids*, Vol. 143, 2017, pp. 1–15. doi:10.1016/j.compfluid.2016.11.002.

[5] Allen, C. B., Poole, D. J., and Rendall, T. C. S., "Wing aerodynamic optimization using efficient mathematically-extracted modal design variables," *Optimization and Engineering*, Vol. 19, No. 2, 2018, pp. 453–477. doi:10.1007/s11081-018-9376-7.

[6] Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[7] Storn, R. and Price, K., "Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, Vol. 11, 1997, pp. 341–359. doi:10.1023/A:1008202821328.

[8] Karaboga, D. and Basturk, B., "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," *Journal of Global Optimization*, Vol. 39, 2007, pp. 459–471. doi:10.1007/s10898-007-9149-x.

[9] Yang, X. S., *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2nd ed., 2010.

[10] Venter, G. and Sobieszczanski-Sobieski, J., "Particle Swarm Optimization," *AIAA Journal*, Vol. 41, No. 8, 2003, pp. 1583–1589.
doi:10.2514/2.2111.

[11] Barbosa, H. J. C. and Lemonge, A. C. C., "A New Adaptive Penalty Scheme for Genetic Algorithms," *Information Sciences*, Vol. 156, No. 3-4, 2003, pp. 215–251.
doi:10.1016/S0020-0255(03)00177-4.

[12] Coello Coello, C. A., "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, Vol. 41, No. 2, 2000, pp. 113–127.
doi:10.1016/S0166-3615(99)00046-9.

[13] Liang, J. J. and Suganthan, P. N., "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism," *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.
doi:10.1109/CEC.2006.1688284.

[14] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "A Generic Framework for Handling Constraints with Agent-Based Optimization Algorithms and Application to Aerodynamic Design," *Optimization and Engineering*, Vol. 18, No. 3, 2017, pp. 659–691.
doi:10.1007/s11081-016-9343-0.

[15] Deb, K., "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, 2000, pp. 311–338.
doi:10.1016/S0045-7825(99)00389-8.

[16] Toscano Pulido, G. and Coello Coello, C. A., "A Constraint handling Mechanism for Particle Swarm Optimization," *2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon, 2004.
doi:10.1109/CEC.2004.1331060.

[17] Sun, C. L., Zeng, J. C., and Pan, J. S., "An Improved Vector Particle Swarm Optimization for Constrained Optimization Problems," *Information Sciences*, Vol. 181, 2011, pp. 1153–1163.
doi:10.1016/j.ins.2010.11.033.

[18] Ward, A., Liker, J. K., Cristiano, J. J., and Sobek II, D. K., "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster," *MIT Sloan Management Review*, Vol. 36, No. 3.

[19] De Jong, K. A., *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, 1975.

[20] Thomsen, R., "Multimodal optimization using crowding-based differential evolution," *Congress on Evolutionary Computation*, Portland, Oregon, 2004.
doi:10.1109/CEC.2004.1331058.

[21] Goldberg, D. E. and Richardson, J., "Genetic algorithms with sharing for multimodal function optimization," *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987.

[22] Petrowski, A., "A clearing procedure as a niching method for genetic algorithms," *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996.
doi:10.1109/ICEC.1996.542703.

[23] Li, J. P., Balazs, M. E., Parks, G. T., and Clarkson, P. J., "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, Vol. 10, No. 3, 2002, pp. 207–234.
doi:10.1162/106365602760234081.

[24] Li, X., "Efficient differential evolution using speciation for multimodal function optimization," *GECCO '05 Proceedings of the 7th annual conference on Genetic and evolutionary computation*, Washington, D.C., 2005.
doi:10.1145/1068009.1068156.

[25] Epitropakis, M. G., Li, X., and Burke, E. K., "A dynamic archive niching differential evolution algorithm for multimodal optimization," *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013. doi:10.1109/CEC.2013.6557556.

[26] Biswas, S., Kundu, S., and Das, S., "Inducing Niching Behavior in Differential Evolution Through Local Information Sharing," *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 2, 2015, pp. 246–263.
doi:10.1109/TEVC.2014.2313659.

[27] Epitropakis, M. G., Plagianakos, V. P., and Vrahatis, M. N., "Finding multiple global optima exploiting differential evolution's niching capability," *2011 IEEE Symposium on Differential Evolution (SDE)*, Paris, France, 2011.
doi:10.1109/SDE.2011.5952058.

[28] Epitropakis, M. G., Plagianakos, V. P., and Vrahatis, M. N., "Multimodal optimization using niching differential evolution with index-based neighborhoods," *2012 IEEE Congress on Evolutionary Computation*, Brisbane, Australia, 2012.
doi:10.1109/CEC.2012.6256480.

[29] Li, X., Epitropakis, M. G., Deb, K., and Engelbrecht, A., "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications," *IEEE Transactions on Evolutionary Computation*, Vol. 21, No. 4, 2017, pp. 518–538.
doi:10.1109/TEVC.2016.2638437.

[30] Chernukhin, O. and Zingg, D. W., "Multimodality and Global Optimization in Aerodynamic Design," *AIAA Journal*, Vol. 51, No. 6, 2013, pp. 1342–1354.
doi:10.2514/1.J051835.

[31] Namgoong, H., Crossley, W., and Lyrintzis, A. S., "Global Optimization Issues for Transonic Airfoil Design," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia, 2002, AIAA Paper 2002-5641.
doi:10.2514/6.2002-5641.

[32] Khurana, M. S., Winarto, H., and Sinha, A. K., "Airfoil Optimisation by Swarm Algorithm with Mutation and Artificial Neural Networks," *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, Florida, 2010, AIAA Paper 2009-1278.
doi:10.2514/6.2009-1278.

[33] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, Vol. 53, No. 4, 2015, pp. 968–985.
doi:10.2514/1.J053318.

[34] Zingg, D. W. and Streuber, G., "Aerodynamic Design Optimization Workshop: Multimodal Subsonic Inviscid Optimization Problem," Tech. rep., AIAA ADODG, 2017, `https://info.aiaa.org/tac/ASG/APATC/AeroDesignOpt-DG/Test%20Cases/ADODG%20Case%206%20Multimodal%20Subsonic%20Inviscid%20Optimization.pdf`.

[35] Du, X., Amrit, A., Thelen, A., Leifsson, L., Zhang, Y., Han, Z.-H., and Koziel, S., "Aerodynamic Design of a Rectangular Wing in Subsonic Inviscid Flow by Surrogate-Based Optimization," *35th AIAA Applied Aerodynamics Conference*, Denver, Colorado, 2017, AIAA Paper 2017-4366.
doi:10.2514/6.2017-4366.

[36] Bons, N. P., He, X., Mader, C. A., and Martins, J. R. R. A., "Multimodality in Aerodynamic Wing Design Optimization," *35th AIAA Applied Aerodynamics Conference*, Denver, Colorado, 2017, AIAA Paper 2017-3753.
doi:10.2514/6.2017-3753.

[37] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "Global Optimization of Wing Aerodynamic Optimization Case Exhibiting Multimodality," *Journal of Aircraft*, Published online.
doi:10.2514/1.C034718.

[38] Streuber, G. M. and Zingg, D. W., "Investigation of multimodality in aerodynamic shape optimization based on the Reynolds-Averaged Navier-Stokes equations," *35th AIAA Applied Aerodynamics Conference*, Denver, Colorado, 2017, AIAA Paper 2017-3752.
doi:10.2514/6.2017-3752.

[39] de Falco, I., Della Cioppa, A., Iazzetta, A., and Tarantino, E., "Evolutionary Algorithms for Aerofoil Design," *International Journal of Computational Fluid Dynamics*, Vol. 11, No. 1-2, 1998, pp. 51–77.
doi:10.1080/10618569808940865.

[40] Giannakoglou, K. C., "Design of Optimal Aerodynamic Shapes Using Stochastic Optimization Methods and Computational Intelligence," *Progress in Aerospace Sciences*, Vol. 38, 2002, pp. 43–76.
doi:10.1016/S0376-0421(01)00019-7.

[41] Epstein, B. and Peigin, S., "Optimization of 3D Wings Based on Navier-Stokes Solutions and Genetic Algorithms," *International Journal of Computational Fluid Dynamics*, Vol. 20, No. 2, 2006, pp. 75–92.
doi:10.1080/10618560600761601.

[42] Blasi, L. and Del Core, G., "Particle Swarm Approach for Finding Optimum Aircraft Configuration," *Journal of Aircraft*, Vol. 44, No. 2, 2007, pp. 679–682.
doi:10.2514/1.24399.

[43] Jahangirian, A. and Shahrokhi, A., "Aerodynamic Shape Optimization Using Efficient Evolutionary Algorithms and Unstructured CFD Solver," *Computers & Fluids*, Vol. 46, 2011, pp. 270–276.
doi:10.1016/j.compfluid.2011.02.010.

[44] Mukesh, R., Pandiyarajan, R., Selvakumar, U., and Lingadurai, K., "Influence of Search Algorithms on Aerodynamic Design Optimisation of Aircraft Wings," *Procedia Engineering*, Vol. 38, 2012, pp. 2155–2163.
doi:10.1016/j.proeng.2012.06.259.

[45] Obayashi, S., Takahashi, S., and Takeguchi, Y., "Niching and elitist models for MOGAs," *International Conference on Parallel Problem Solving from Nature*, 1998.
doi:10.1007/BFb0056869.

[46] Storn, R. and Price, K., "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. rep., ICSI, UC Berkeley, 1995, TR-95-012.

[47] Das, S. and Suganthan, P. N., "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, 2011, pp. 4–31.
doi:10.1109/TEVC.2010.2059031.

[48] Neri, F. and Tirronen, V., "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, Vol. 33, 2010, pp. 61–106.
doi:10.1007/s10462-009-9137-2.

[49] Das, S., Mullick, S. S., and Suganthan, P. N., "Recent advances in differential evolution - An updated survey," *Swarm and Evolutionary Computation*, Vol. 27, 2016, pp. 1–30.
doi:j.swevo.2016.01.004.

[50] Datta, R. and Deb, K., eds., *Evolutionary Constrained Optimization*, Springer India, 2015.
doi:10.1007/978-81-322-2184-5.

[51] Mezura-Montes, E., Coello Coello, C. A., and Tun-Morales, E. I., "Simple Feasibility Rules and Differential Evolution for Constrained Optimization," in "MICAI 2004: Advances in Artificial Intelligence," Springer, Berlin, 2004.
doi:10.1007/978-3-540-24694-7_73.

[52] Qu, B. Y., Suganthan, P. N., and Liang, J. J., "Differential Evolution With Neighborhood Mutation for Multimodal Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 5, 2012, pp. 601–614.
doi:10.1109/TEVC.2011.2161873.

[53] Wilcoxon, F., "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, Vol. 1, No. 6, 1945, pp. 80–83.
doi:10.2307/3001968.

[54] Jones, R. T., "The Spanwise Distribution of Lift for Minimum Induced Drag of Wings Having a Given Lift and a Given Bending Moment," Tech. rep., NACA, 1950, NACA Report 2249.

[55] Klein, A. and Viswanathan, S. P., "Approximate Solution for Minimum Induced Drag of Wings with Given Structural Weight," *Journal of Aircraft*, Vol. 12, No. 2, 1975, pp. 124–126.
doi:10.2514/3.44425.

[56] Katz, J. and Plotkin, A., *Low-Speed Aerodynamics*, Cambridge University Press, 2001.
doi:10.1017/CBO9780511810329.

[57] Zhou, J. L., Tits, A. L., and Lawrence, C. T., "Users Guide for FSQP Version 3.7 : A Fortran Code for Solving Optimization Programs, Possibly Minimax, with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates," Tech. rep., Institute for Systems Research, University of Maryland, 1997, SRC-TR-92-107r5.

[58] Mayne, D. Q. and Polack, E., "A surperlinearly convergent algorithm for constrained optimization problems," *Mathematical Programming Studies*, Vol. 4, 1982, pp. 45–61.
doi:10.1007/BFb0120947.

[59] Grippo, L., Lampariello, F., and Lucidi, S., "A Nonmonotone Line Search Technique for Newton's Method," *SIAM Journal on Numerical Analysis*, Vol. 23, No. 4, 1986, pp. 707–716.
doi:10.1137/0723046.

[60] Panier, E. R. and Tits, A. L., "Avoiding the Maratos Effect by Means of a Nonmonotone Line Search I. General Constrained Problems," *SIAM Journal on Numerical Analysis*, Vol. 28, No. 4, 1991, pp. 1183–1195.
doi:10.1137/0728063.

[61] Bonnans, J. F., Panier, E. R., Tits, A. L., and Zhou, J. L., "Avoiding the Maratos Effect by Means of a Nonmonotone Line Search II. Inequality Constrained Problems–Feasible Iterates," *SIAM Journal on Numerical Analysis*, Vol. 29, No. 4, 1992, pp. 1187–1202.
doi:10.1137/0729072.